EE414 Embedded Systems

Ch 3. General-Purpose Processors: Software

Part 4/4: Linux Development Environment

Byung Kook Kim
School of Electrical Engineering
Korea Advanced Institute of Science and Technology

Overview

Linux Development Environment

- 3.41 Operating System
- 3.42 Embedded Linux
- 3.43 Cross-Development System
- 3.44 Setting Linux Development Environment
- 3.45 Linux Basics

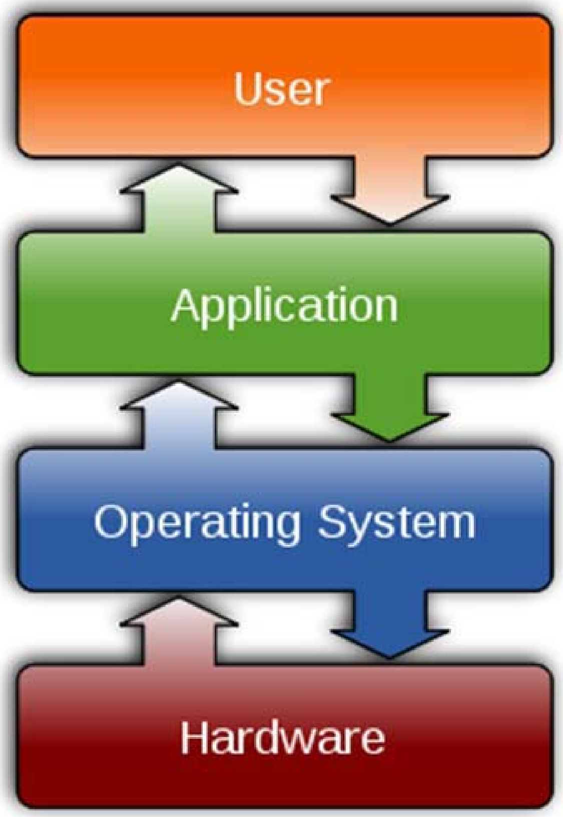
3.41 What is Operating System?

- An **operating system** (**OS**) is a set of software that manages computer hardware resources and provides common services for computer programs. The operating system is a vital component of the system software in a computer system. Application programs require an operating system to function.
- Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting for cost allocation of processor time, mass storage, printing, and other resources.
- For hardware functions such as input and output and memory allocation, the operating system acts as an intermediary between programs and the computer hardware, although the application code is usually executed directly by the hardware and will frequently make a system call to an OS function or be interrupted by it. Operating systems can be found on almost any device that contains a computer—from cellular phones and video game consoles to supercomputers and web servers.
- Examples of popular modern operating systems include Android, BSD, iOS, Linux, Mac OS X, Microsoft Windows, Windows Phone, and IBM z/OS. All these, except Windows and z/OS, share roots in UNIX.

Ref. http://en.wikipedia.org/wiki/Operating_system

Common features of OS

- Process management
- Interrupts
- Memory management
- File system
- Device drivers
- Networking (TCP/IP, UDP)
- Security (Process/Memory protection)
- I/O



Linux

- **Linux** originated by Linus Torvalds
  - Vast application software
  - Stability of kernels
  - Success in servers and workstations using PC

- Advantages of Linux
  - Compatible with Unix
  - Open source, free
  - Stable (than Windows)
  - Improved hardware utilization
  - Powerful networking and Internet support
  - Vast application programs
  - Multi-user, multi-tasking
  - Supports POSIX (Portable Operating System Interface for Computer Environment).

Ubuntu

- Super-fast, easy to use and free, the Ubuntu Linux operating system powers millions of desktops, netbooks and servers around the world.
  - Ubuntu does everything you need it to.
  - It'll work with your existing PC files, printers, cameras and MP3 players.
  - And it comes with thousands of free apps.

- Works on PC, Notebook.
- Works on BeagleBone also (without Graphic User Interface)

Ref. http://www.ubuntu.com/ubuntu

Debian

- Debian is a free operating system (OS) for your computer. An operating system is the set of basic programs and utilities that make your computer run.
  - Debian provides more than a pure OS: it comes with over 43000 packages, precompiled software bundled up in a nice format for easy installation on your machine.

- **Getting Started**
  - If you'd like to start using Debian, you can easily obtain a copy, and then follow the installation instructions to install it.
  - www.debian.org
  - Works on BeagleBone also (without Graphic User Interface)

Ref. http://www.ubuntu.com/ubuntu

3.42 Embedded Linux

- **Software for embedded target**
  - Simple systems (Ex. Automatic electric power meter)
    - Application software, modem communication software

  - More complex systems (Ex. PDA)
    - Application/service software
      - Personal data management, game, electronic commerce, remote control of home appliance, cellularphone, Web surfing, chatting, etc.
    - Hardware dependent platform driving software
      - Software driving internal and external hardware
      - Device driver for LCD, keypad, touch panel, network, etc.
    - Network software
      - xDSL, cable modem, Ethernet, Bluetooth, Wifi, CDMA, etc.
    - Fundamental software
      - Operating system, DBMS, GUI, MMI, Web server

Embedded Linux (II)

- **Operating system for embedded systems**
  - Why OS?
    - Complex programs and diverse services (network and devices)
    - Fast development time and expandability
    - OS: Program that manages computer hardware and software resources efficiently.
    - ~60% of embedded systems utilize OS.
  - Constraints
    - Should fit within system's memory
      - RAM: data
      - ROM or flash: program
  - Operating systems
    - Wind River Systems: VxWorks, Tornado
    - Palm computing: Palm OS
    - Microsoft: Windows CE
    - **-: Embedded Linux - Next OS to be applied (49%)**

Embedded Linux (III)

- **Embedded Linux**
  - ***Scaled-down Linux for embedded low-performance processors to fit into small ROM or flash.***

  - With or without memory management software (or virtual memory)
  - Ported processors
    - 32bit: Intel x86, Motorola Power PC, ARM9, MIPS, etc.
    - 64-bit: IA-64
    - W/O MMU: ARM7, Motorola 68K, Intel i960, AXIS, etc.

3.43 Cross-Development System

- **Embedded system software**
  - Whole software necessary to develop embedded system
    - User: Software running on the embedded system
    - Developer: Software for cross development environment

- **Software development tools**
  - Editor: Edit source files
  - Compiler: Translates into object files
  - Linker: Links object files and libraries
  - Debugger: Step-by-step execution and status check

Cross-Development System (II)

- **Stand-alone system**
  - PC and Workstation
  - Self-contained for general purposes
    - Hardware: CPU. Memory, general-purpose user interface, Disk,
    - Software: Operating system, application program, editor, compiler, linker
  - Native compiler
    - Compiler for the program being developed using the PC and being run on the PC.
- **Embedded system**
  - Self-contained for a specific purpose
    - Limited hardware: CPU, memory, specific I/O, Flash or ROM
    - Software: Embedded OS, application program
  - Cross compiler
    - Compiler for the target processor, but running on the development computer (PC). Compiled program should be downloaded to the target processor and then be executed.

Cross-compiler

- A compiler capable of creating executable code for a platform other than the one on which the compiler is run.
- Uses of cross compilers: The fundamental use of a cross compiler is to separate the build environment from target environment.
  - Embedded computers where a device has extremely limited resources.
  - This computer will not be powerful enough to run a compiler, a file system, or a development environment.
  - Since debugging and testing may also require more resources than are available on an embedded system, cross-compilation can be less involved and less prone to errors than native compilation.

[http://en.wikipedia.org/wiki/Cross_compiler]

Cross-Development

- **Development stages for embedded system**
  - *In the host*
    - 1. Design and development of hardware-independent software
      - Compile, run, and debug on the host
    - 2. Cross-compile into executable code on the target embedded processor
    - 3. Download the executable code to the embedded target
  - *In the target (with the host)*
    - 4. Run and debug using debugging tools
    - 5. Transfer the verified program to ROM or flash in the embedded target.

3.44 Setting Linux Development Environment

- **Partitioning Disk**
  - Windows requires at least one disk partition (C:)
    - Add one more partition for user space (D:)
  - Linux requires at least two disk partitions (/ and swap)
    - Add one more partition for user space (/home)
  - Up to four primary partitions and four logical partitions inside one partition.
  - Solution
    - Primary partition: C: for Windows
    - Secondary partition: D: for windows
    - Third partition: Up to 4 logical partitions
      - 1st partition: / for Linux (16 GB or more)
      - 2nd partition: Swap for Linux (2x memory size)
      - 3rd partition: /home for Linux (32 GB or more).

Install Ubuntu 16.04 for dual boot

Refer http://www.ubuntu.com/download/help/install-ubuntu-desktop

- It's easy to install Ubuntu from a CD. Here's what you need to do:
  - Put the Ubuntu CD into the CD/DVD-drive.
  - Restart your computer.
- After a while, Ubuntu logo will be displayed at the center of display, and then "Install – Welcome" window appears.
  - In the left column, select "English"
  - In the right column, select "Install Ubuntu".
- In the "Install – Preparing" window, select "continue".
- IMPORTANT: In the 'Install – Installation Type" window, Select "Something Else" in order to make ***Linux partitions***.
  - Click "Add", and create a new partition. You should make three partitions:

/dev/sdb2	ext4 file	/	16 GB or more
/dev/sdb3	swap		Twice of main memory size
/dev/sdb4	ext4 file	/home	32 GB or more

Install Ubuntu 16.04 for dual boot (II)

- In “Install – Where are you?” window, select your time zone, e.g., “Seoul”.
- In “Install – Keyboard Layout” window, select left “Korean”, and right “Korean”.
- In “Install – Who are you?” window, set your name, id, and password (twice).
- Wait a while…

- When the installation is complete, select “Restart Now” button.
- Remove Ubuntu 16.04 Cd from CD driver, and press “Enter” key.
- Ubuntu 16.04 will boot!

Cross Compiler Setup

- Check existence of cross-compiler in PC Ubuntu.

```
$ arm-linux-gnueabihf-gcc --version
arm-linux-gnueabihf-gcc (Ubuntu/Linaro 4.6.2-14ubuntu2~ppa1)
4.6.2
Copyright (C) 2011 Free Software Foundation, Inc.
……
```

- If none, install cross-gcc

```
$ sudo apt-get install gcc-arm-linux-gnueabihf
```

- Check the location of cross-gcc

```
$ sudo find / -name arm-linux-gnueabihf-gcc -print
[sudo] password for bkkim:
/usr/bin/arm-linux-gnueabihf-gcc
```

NFS Environment

- **NFS (Network File System)**
  - Enables the user to access files in the remote-host as if it is local, using RPC.
  - Combined effort of the remote host file system with NFS server, and client's kernel.
  - Useful on various servers and host architectures.

- Adv.
  - No download is required to run on the target.
  - Very large files (larger than RAMDISK) can also be executed.
  - Many developers can share software programs on a development PC.

- Caution
  - Special files (e.g., device file) cannot be connected via nfs.
  - Fast read/write files (e.g., multimedia files) cannot be used.

Set up NFS server on PC Ubuntu

- Edit /etc/exports.
  - $ sudo vi /etc/exports

```
# /etc/exports
# Directory information to export via NFS
/home/bkkim/u-bone-ubuntu
          192.168.0.6(rw,sync,no_root_squash)
```

- Whenever we modify /etc/exports, we must run

  ```
  $ sudo exportfs -a
  ```

  - It makes the changes effective afterwards.
- Starting the Portmapper

  ```
  $ ps -aux | grep portmap
  ```

- Check if nfs started.

  ```
  $ ps aux | grep nfs
  ```

Set NFS client on BeagleBone Ubuntu

- Install nfs client in Bone
  - # sudo apt-get nfs-common
- Make directory for nfs
  - # mkdir /mnt/es
- Start nfs
  - # cd ~
  - # mount -t nfs -o nolock 192.168.0.5:/home/bkkim/embedded12-bone-ubuntu /mnt/es
- For later use, edit ~/es-nfs

```
#! /bin/sh
# Mount nfs in Bone-Ubunut
echo "Mount nfs in Bone-Ubuntu
/mnt/bu..."
mount -t nfs -o nolock
192.168.0.5:/home/bkkim/embedded12-
bone-ubuntu /mnt/es
```

- Change mode of bu-nfs to 755
  - # chmod 755 bu-nfs
- Later, you can use simplay
  - # ~/es-nfs
- Go to nfs directory
  - # cd /mnt/es
  - # ls
- You can see PC-Ubuntu directory ~/embedded12-bone-ubuntu!

System Software

Software	Development PC	Embedded Board
App	-	Hello_es
Terminal	Terminal	Terminal with screen cmd
NFS	NFS Server	NFS client
Secure copy	Scp send	Scp receive
Compiler	Cross-compiler for ARM	-
Editor	gedit	-
OS	Linux	Linux

3.45 Linux Basics

- Getting Started
  - # login: *username* or *root*
  - # password: *user_password* or *root_password*
  - # logout
  - # shutdown –h now ; Shutdown the computer
- Basic commands
  - # date ; Display date and time
    - Wed Sep 1 12:12:29 EDT 2004
  - # who ; List users currently logged in
  - # man command ; Display manual of the command
  - # pwd ; Print the complete pathname of the current directory
  - # cd /usr/src/linux ; Change directory to /usr/src/linux

Linux Basics (II)

- File manipulation
  - # ls [-la] ; List files in the current directory
  - # cat filename ; Prints the file with filename
  - # cp source_file dest_file ; Copy source_file to dest_file
    - # cp file /dev/ttyS0 ; Copy file to COM1
  - # rm junk_file ; Remove junk_file
  - # mv old_file to new_file ; Rename the old_file to new_file

- Manipulating directories
  - # mkdir new_dir ; Make a new_dir directory
  - # rmdir old_dir ; Delete the old_dir directory
  - # mv old_dir new_dir ; Rename old_dir directory to new_dir
  - # cd new_dir ; Change directory to new_dir
    - # cd .. ; Change to upper directory
    - # cd / ; Change to root directory

Linux Basics (III)

- System inquiries

# ps	; List active processes with process_id
# kill -9 process_id	; Kill the process with process_id
# du	; Disk usage of the current directory
# df	; Display file system usage
# su	; Become the superuser (root)
# password: *root_password*	
# exit	; Become a normal user

Linux Basics (IV)

- Editing files with vi ; Keyboard only. No mouse.
  - # vi file.c ; Visual edit file.c
  - # Ctrl-F, Ctrl-B : Move forward/backward a full screen
  - # space, backspace, return ; Move cursor right/left/next_line
  - # i… esc ; Insert characters before cursor (until escape)
  - # a… esc ; Insert characters after cursor (until escape)
  - # o… esc ; Insert line by line after the current line
  - # O… esc ; Insert line by line before the current line
  - # x ; Delete the current character
  - # dw ; Delete the current word
  - # dd ; Delete the current line
  - # r file ; Read the file
  - # s/old/new/g ; Substitute old to new globally
  - # :q ; Quit without saving
  - # :wq ; Quit after saving

Linux Basics (V)

```
#include <stdio.h>
void main()
{
        printf("Hello, Embedded system!₩n");
}
```

- Compile and run
  - # mkdir /embedded/test
  - # cd /embedded/test
  - # vi hello.c
  - # gcc –o hello hello.c

    ; Native-compile and link the program to produce hello.
  - # ./hello

    ; Run hello
  - Hello, Embedded board!

    ; Output: print a string on the console

Linux Basics (VI)

- Make command
  - # vi Makefile

```
main.o average.o: defs.h
average: main.o average.o
        gcc –o average main.o average.o –lm
```

  - # vi defs.h
  - # vi main.c
  - # vi average.c
  - # make average.o ; Compile average.c to average.o
  - # make average ; Compile main.c to main.o
    Link main.o, average.o, and lib into average
  - # ./average ; Run average

Reference

- Search Internet…